US007596665B2

(12) **United States Patent**
Day et al.

(10) **Patent No.: US 7,596,665 B2**
(45) **Date of Patent: *Sep. 29, 2009**

(54) **MECHANISM FOR A PROCESSOR TO USE LOCKING CACHE AS PART OF SYSTEM MEMORY**

(75) Inventors: **Michael Norman Day**, Round Rock, TX (US); **Charles Johns**, Austin, TX (US); **Thuong Truong**, Austin, TX (US)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/874,513**

(22) Filed: **Oct. 18, 2007**

(65) **Prior Publication Data**

US 2008/0040548 A1 Feb. 14, 2008

Related U.S. Application Data

(63) Continuation of application No. 10/976,260, filed on Oct. 28, 2004, now Pat. No. 7,290,106.

(51) **Int. Cl.**
G06F 12/08 (2006.01)

(52) **U.S. Cl.** **711/129**; 711/144; 711/145

(58) **Field of Classification Search** 711/129, 711/128, 141, 146, 145, 144
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,643,737	B1 *	11/2003	Ono	711/128
6,859,862	B1 *	2/2005	Liao et al.	711/129
7,120,651	B2 *	10/2006	Bamford et al.	711/129
2001/0001873	A1	5/2001	Wickeraad et al.	
2002/0046326	A1	4/2002	Devereux	
2002/0046334	A1	4/2002	Wah Chan et al.	
2002/0062424	A1 *	5/2002	Liao et al.	711/129
2002/0065992	A1	5/2002	Chauvel et al.	
2006/0095668	A1	5/2006	Day et al.	
2006/0095669	A1	5/2006	Day et al.	

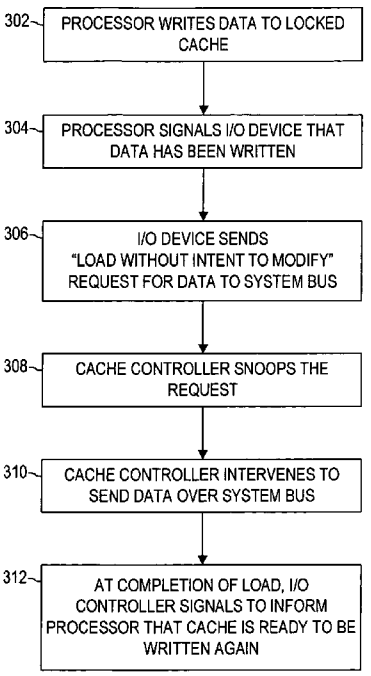
* cited by examiner

Primary Examiner—Pierre-Michel Bataille
(74) *Attorney, Agent, or Firm*—Francis Lammes; Stephen J. Walder, Jr.; Matthew B. Talpis

(57) **ABSTRACT**

The present invention provides a mechanism for a processor to write data to a cache or other fast memory, without also writing it to main memory. Further, the data is "locked" into the cache or other fast memory until it is loaded for use. Data remains in the locking cache until it is specifically overwritten under software control. The locking cache or other fast memory can be used as additional system memory. In an embodiment of the invention, the locking cache is one or more sets of ways, but not all of the sets or ways, of a multiple set associative cache.

18 Claims, 4 Drawing Sheets



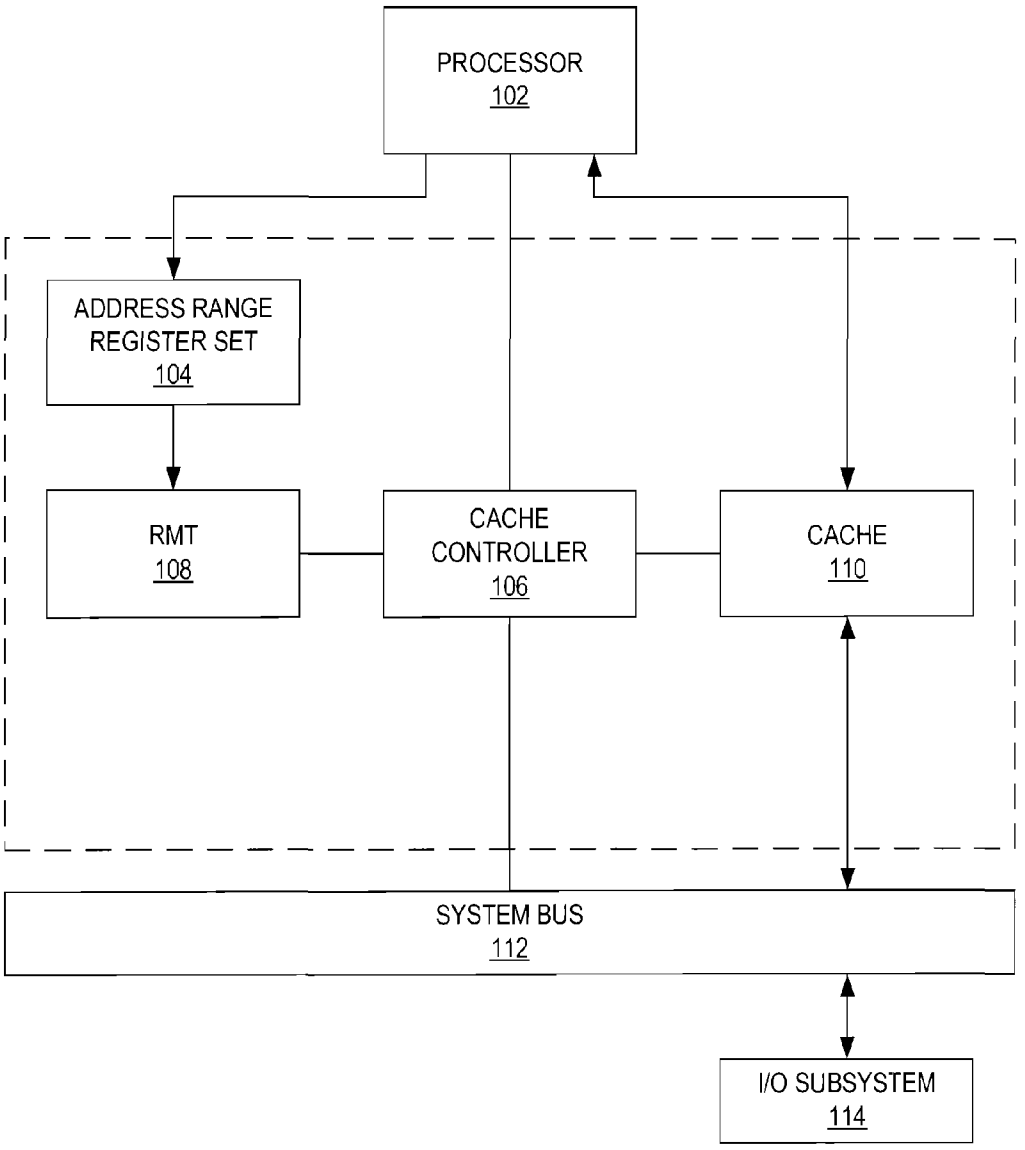

PROCESSOR
102
ADDRESS RANGE
REGISTER SET
104
RMT
108
CACHE
CONTROLLER
106
CACHE
110
SYSTEM BUS
112
I/O SUBSYSTEM
114

FIG. 1

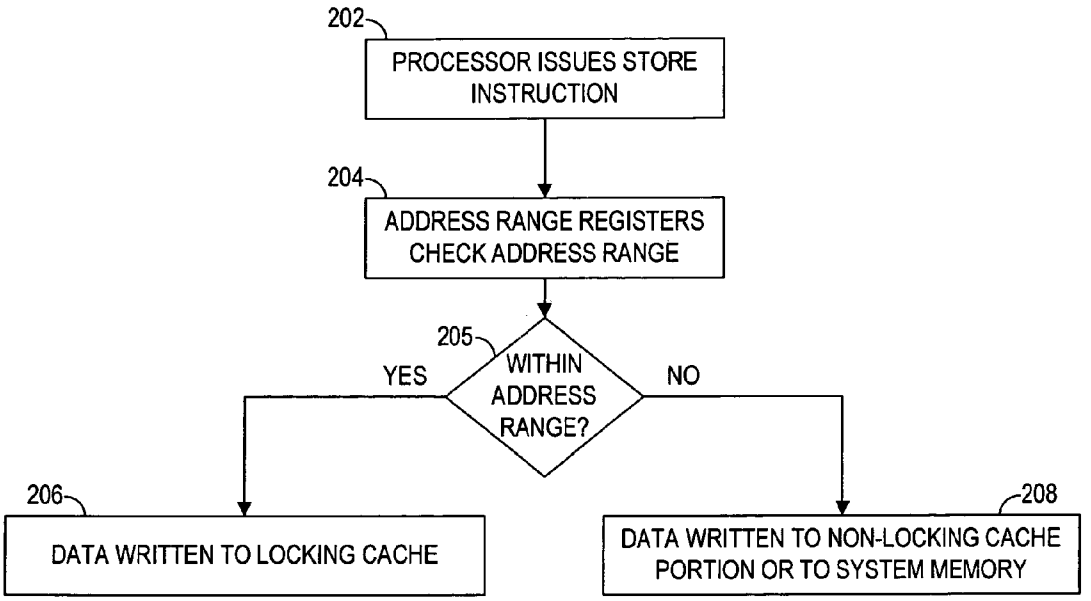

202
PROCESSOR ISSUES STORE INSTRUCTION
204
ADDRESS RANGE REGISTERS CHECK ADDRESS RANGE
205
WITHIN ADDRESS RANGE?
YES
NO
206
DATA WRITTEN TO LOCKING CACHE
208
DATA WRITTEN TO NON-LOCKING CACHE PORTION OR TO SYSTEM MEMORY

FIG. 2

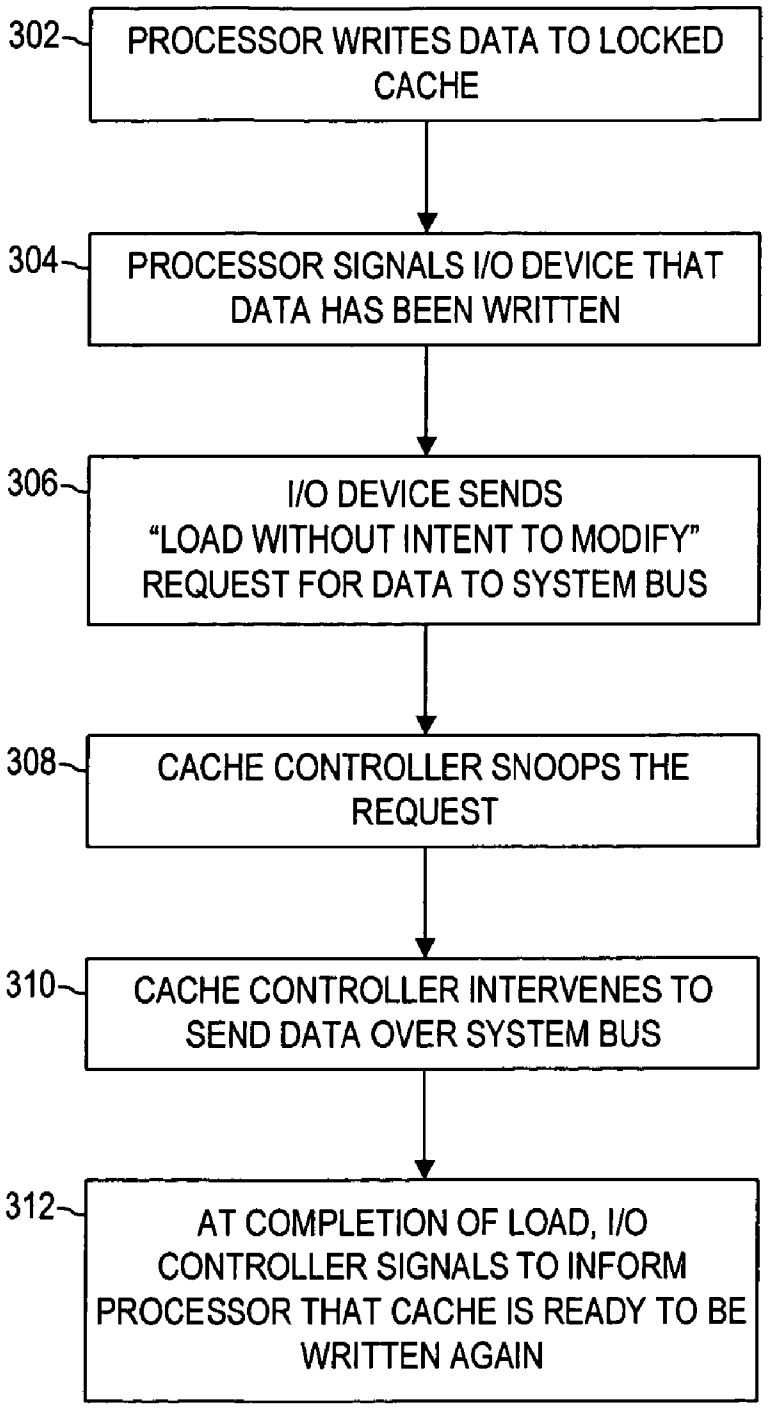

302
PROCESSOR WRITES DATA TO LOCKED CACHE
304
PROCESSOR SIGNALS I/O DEVICE THAT DATA HAS BEEN WRITTEN
306
I/O DEVICE SENDS "LOAD WITHOUT INTENT TO MODIFY" REQUEST FOR DATA TO SYSTEM BUS
308
CACHE CONTROLLER SNOOPS THE REQUEST
310
CACHE CONTROLLER INTERVENES TO SEND DATA OVER SYSTEM BUS
312
AT COMPLETION OF LOAD, I/O CONTROLLER SIGNALS TO INFORM PROCESSOR THAT CACHE IS READY TO BE WRITTEN AGAIN

FIG. 3

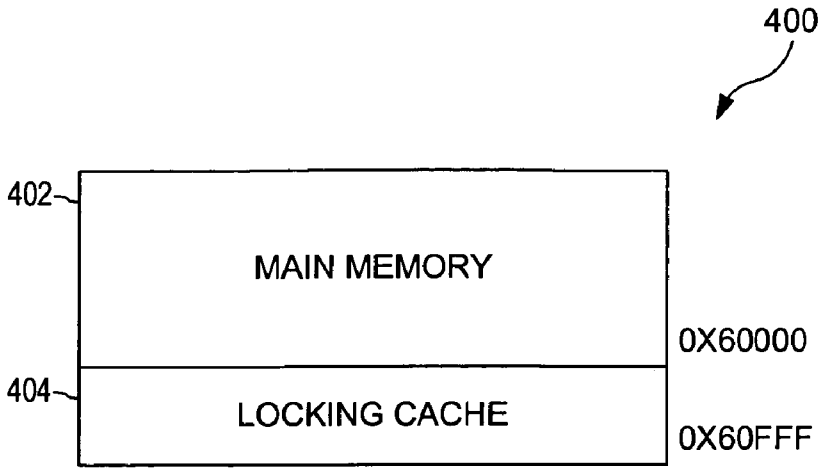


FIG. 4

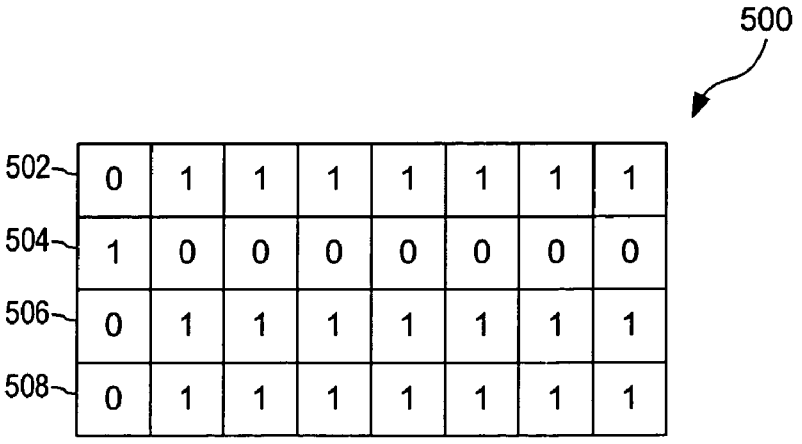


FIG. 5

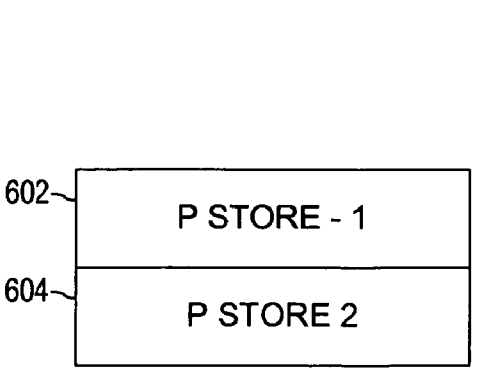


FIG. 6

MECHANISM FOR A PROCESSOR TO USE LOCKING CACHE AS PART OF SYSTEM MEMORY

This application is a continuation of application Ser. No. 10/976,260, filed Oct. 28, 2004, now U.S. Pat. No. 7,290,106.

RELATED APPLICATIONS

This application relates to a co-pending U.S. patent application entitled "Direct Deposit Using Locking Cache" U.S application Ser. No. 10/976,263 in the names of Michael Norman Day, Charles Ray Johns, and Thuong Quang Truong, filed concurrently herewith.

TECHNICAL FIELD

The present invention relates generally to memory management and, more particularly, to the use of caches.

BACKGROUND

The latency (time spent waiting) for memory access, both to write to memory and to read from memory, is often a problem for software programs. In current computers, processor cycles are much shorter than the time for memory access. Further, the problem is becoming more severe. Processor speed is increasing exponentially, and memory access is increasing only gradually.

One partial remedy to the problem of memory access latency is a hierarchy of memories. The main memory has a large capacity and is slowest. On top of this are several layers of successively smaller, faster memories, or caches.

The current use of caches presents problems. A read from a cache may fail when the cache does not contain the desired data. The data must then be accessed from the slow main memory. An attempt to write data exclusively to a cache may not be permitted. Data from the processor can be written to the cache and then pushed to main memory. Thus, there is the latency of writing to the slower main memory. Further, there can be a latency in accessing the data. The data written to a cache may be replaced by other data before the replaced data is accessed. When this occurs, the replaced data is written to main memory. To then utilize this data, the data must be accessed from main memory.

Therefore, there is a need for a method for a processor to write data to a cache or other fast memory without also writing it to main memory. Further, the method must guarantee that the data remains in the cache or other fast memory until it has been used.

SUMMARY OF THE INVENTION

The present invention provides a method for a processor to write data to a cache or other fast memory, without also writing it to main memory. Further, the data is "locked" into the cache or other fast memory until it is loaded for use. Data remains in the locking cache until it is specifically overwritten under software control.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawing, in which:

FIG. **1** shows a block diagram of a system for a processor to write data to a locking cache;

FIG. **2** shows a flow diagram illustrating the address range check when a processor stores data;

FIG. **3** shows a flow diagram illustrating a processor writing data to a locking cache;

FIG. **4** is a diagram showing the layout of memory from the perspective of a processor;

FIG. **5** illustrates a replacement management table; and

FIG. **6** illustrates a partitioning of the locking cache.

DETAILED DESCRIPTION

In the following discussion, numerous specific details are set forth to provide a thorough understanding of the present invention. However, it will be apparent to those skilled in the art that the present invention may be practiced without such specific details. In other instances, well-known elements have been illustrated in schematic or block diagram form in order not to obscure the present invention in unnecessary detail.

It is further noted that, unless indicated otherwise, all functions described herein may be performed in either hardware or software, or some combination thereof. In a preferred embodiment, however, the functions are performed by a processor such as a computer or an electronic data processor in accordance with code such as computer program code, software, and/or integrated circuits that are coded to perform such functions, unless indicated otherwise.

FIG. **1** shows a block diagram of a system for a processor **102** to write data to a locking cache. The processor **102** is coupled to a cache **110**, a cache controller **106**, and a set of address range registers **104**. A replacement management table (RMT) **108** is coupled to the address range register set **104** and to the cache controller **106**. The cache controller **106** and the cache **110** are coupled to a system bus **112**. The system bus **112** is further coupled to an input/output (I/O) subsystem **114**. In an embodiment of the invention, the locking cache comprises a set or sets, but not all of the sets, of a multiple set-associative cache **110**. The remaining sets are used for regular cache. The separation of the cache **110** into disjoint sets for use as regular cache and locking cache prevents data written to the locking cache from being overwritten by data written to the cache in its normal use.

To the processor **102**, space in the locked cache appears as additional system memory, with an address range higher than actual main system memory address range. In an embodiment of the invention, a set of address range registers **104** determines access to the locked cache. The set includes two address range registers and a mask register. The accessing address of a load or store instruction is compared to the content of address range registers. A class_id is then provided as an index into replacement management table (RMT) **108**. The RMT **108** indicates which sets of the cache **110** are available to the load or store instruction. Transactions whose accessing address is within the specified range have access to the locking cache. Data written to the locking cache will remain there until overwritten under software command. It can be kept in the locking cache until it is loaded for use.

As a result of the system of FIG. **1**, the processor **102** can write newly generated data to the locking cache, a fast form of memory, rather than to the much slower main memory. Further, both the processor **102** and I/O subsystem **114** can load data from the locking cache, avoiding the latency of loading it from main memory. The data is initially written to the locking cache, and is locked into the cache **110** until it is accessed for use.

FIG. **2** shows a flow diagram illustrating the address range check when a processor stores data. In step **202**, the processor **102** issues a store request. In step **204**, a pair of address range registers in the address range register set **104** checks the address range of the request. In an embodiment of the invention, the address range register set **104** can also contain a masking register. In step **205**, it is determined whether the address of the request is within range. If the address is within range for the locking cache, then in step **206**, the data is written to the locking cache. If the address is not within range, then in step **208**, the data is written to the non-locking portion of the cache or to system memory. In an embodiment of the invention, in step **208** the data is written to system memory. In another embodiment of the invention, in step **208** the data is written both to system memory and to the cache **110**, but not to the portion of the cache used for the locking cache. In yet another embodiment of the invention, in step **208** the data is written to the cache **110**, but not to the portion used for the locking cache.

FIG. **3** shows a flow diagram illustrating the storing of data in the locking cache and the accessing of the data by the I/O subsystem **114**. In step **302**, the processor **102** writes the data to the locking cache. In step **304**, the processor signals the I/O subsystem **114** that the data has been written. Once notified by the signal, in step **306**, the I/O device sends a "load without intent to modify" request for data to the system bus **112**. In the locked portion of the cache, data is marked valid and modified. When an IO controller or other device accesses this data, the IO controller or other device loading the data issues a load without intent to modify request. Data in this address range is stored from the processor **102** without the need of a bus transaction because of the "valid and modified" cache state.

In step **308**, the cache controller **106** snoops the request. Given the state of the data in the cache **110**, in step **310**, the cache controller **106** intervenes to send the data over the system bus **112**. Similarly, when data in the address range of the locked cache is loaded by the processor **102**, the cache controller **106** returns the data to the processor **102** as a cache hit. In step **312**, at the completion of the load, the I/O controller signals to inform the processor **102** that the locking cache is ready to be written again. The space holding the data is then available for further writing. To insure the validity of data, an area of the locked cache to which data is being written by the processor **102** is not simultaneously being read or written to by the processor or another device.

FIG. **4** is a diagram showing the layout of memory **400** from the perspective of a processor. The locking cache appears to be additional system memory with an address range above that of the main system memory. In FIG. **4**, main memory **402** ends with address 0X60000 (hex), and the locking cache **404** contains addresses 0X60001 (hex) through 0X60FFF (hex). The locking cache illustrated in FIG. **4** contains 4 kb. The size of the locking cache is implementation dependent. Although the main memory and locking cache address spaces are consecutive in FIG. **4**, in other embodiments, the address spaces do not have to be consecutive.

FIG. **5** illustrates a replacement management table (RMT) **500** having four rows of entries, **502**, **504**, **506**, and **508**, each row being indexed by the binary numbers 00, 01, 10, and 11, respectively. The entries in a row of the RMT **500** indicate which sets in a cache are available for a transaction. Columns correspond to the ways or sets of the cache **110**. A 1-bit in a column designates that the corresponding way is available to the transaction, and a 0-bit designates that the corresponding way is not available. Transactions involving the locked cache **404** are provided a class_id that gives an index into a row with 1's for the sets comprising the locked cache and 0's for the other sets. Transactions not involving the locked cache are provided a class_id that gives an index into a row with 0's for the sets comprising the locked cache and a 1 for at least one set in the cache not involving the locked cache. The cache corresponding to the RMT in FIG. **5** has eight sets or ways. The first set is used as the locking cache, and the remaining sets are used for regular cache. There are four rows to the RMT. The index 01, corresponding to the second row **504**, is used for transactions which access the locking cache. The "1" in the first column of the row **504** indicates that the first set, the one used for the locking cache, is available for the transaction. The "0"s in the remaining columns of the row **504** indicate that the other sets in the cache are not available for the transaction. The other rows **502**, **506**, and **508** indicate that the set used for the locking cache is not available, but the sets comprising the normal cache are available.

In other embodiments, multiple sets can be used for the locking cache. In those embodiments, software selects the set in which to store particular data. The software can begin writing to the first set of the locking cache. When that set is filled up, the software can begin to write to the second set of the locking cache.

FIG. **6** illustrates a partitioning of the locking cache **404** into two partitions or segments. The processor **102** can write data to the second segment **604** while it is waiting for the I/O subsystem **114** to access data that has been written to the first segment **602**. Similarly, the processor **102** can write data to the first segment **602** while it is waiting for the I/O subsystem **114** to access data that has been written to the second segment **604**. Thus, the processor **102** can avoid the latency of waiting for data to be accessed before storing other data.

Having thus described the present invention by reference to certain of its preferred embodiments, it is noted that the embodiments disclosed are illustrative rather than limiting in nature and that a wide range of variations, modifications, changes, and substitutions are contemplated in the foregoing disclosure and, in some instances, some features of the present invention may be employed without a corresponding use of the other features. Many such variations and modifications may be considered desirable by those skilled in the art based upon a review of the foregoing description of preferred embodiments. Accordingly, it is appropriate that the appended claims be construed broadly and in a manner consistent with the scope of the invention.

The invention claimed is:

1. A method for writing data from a processor of a computer system directly to a locking cache and for retaining the data in the locking cache until accessed for use, the method comprising:

partitioning, by the processor, a cache into a locking cache and a non-locking cache;

receiving, by a cache controller, a store instruction from the processor;

determining, by the cache controller, whether the store instruction is intended for the locking cache;

configuring, by the processor, the locking cache so that data written to the locking cache will not be overwritten by at least one application running on the processor until the data is loaded for use;

if the store instruction is intended for the locking cache, writing, by the processor, data to the locking cache;

responsive to the data being written to the locking cache, issuing, by the processor, a first signal to an I/O subsystem indicating that the data in the locking cache is ready for use by the I/O subsystem;

loading, by the I/O subsystem, the data for use; and

responsive to loading the data for use, issuing, by the I/O subsystem, a second signal to the processor indicating that the data in the locking cache can be overwritten by the at least one application running on the processor.

2. The method of claim **1**, further comprising the steps of:

partitioning the locking cache;

writing data to a second partition after the completion of writing data to a first partition.

3. The method of claim **1**, wherein the locking cache comprises one or more sets or ways of a multiple set-associative cache, but not all of the sets or ways.

4. The method of claim **3**, wherein a replacement management table is used to indicate which set or sets of the cache is accessible to a transaction.

5. The method of claim **1**, wherein the step of configuring the locking cache further comprises marking the data in the locking cache as "valid and modified".

6. The method of claim **5**, wherein the step of loading the data for use comprises the steps of:

issuing a load request without intent to modify from an I/O subsystem of the computer system;

snooping the request; and

intervening to transmit the data from the cache over a system bus.

7. The method of claim **1**, wherein the computing system comprises one or more address range registers that store a specified address range to access the locking cache such that the locking cache appears as additional system memory, and wherein determining whether the store instruction is intended for the locking cache comprises determining whether the store instruction is within the specified address range for the locking cache.

8. A computer system, comprising:

a processor;

a cache connected to the processor, wherein the cache is partitioned into a locking cache and a non-locking cache;

a cache controller connected to the processor and the cache;

a system bus connected to the cache;

an I/O subsystem connected to the system bus;

wherein the cache controller is configured to receive a store instruction from the processor and determine whether the store instruction is intended for the locking cache;

wherein the processor is configured to write data to the locking cache without also writing it to system memory if the store instruction is intended for the locking cache;

wherein the processor is configured to send a first signal to the I/O subsystem, responsive to the data being written to the locking cache, indicating that the data in the locking cache is ready for use by the I/O subsystem;

wherein the locking cache is at least configured to retain the data until the data has been accessed for use by the I/O subsystem; and

wherein the processor is configured to receive a second signal from the I/O subsystem, responsive to the data being loaded for use, indicating that the data in the locking cache can be overwritten by the at least one application running on the processor.

9. The computer system of claim **1**, further configured for load and store instructions with a specified address range to access the locking cache.

10. The computer system of claim **1**, further comprising one or more address range registers that store a specified address range to access the locking cache such that the locking cache appears as additional system memory, wherein the cache controller is configured to determine whether the store instruction is within the specified address range for the locking cache.

11. The computer system of claim **1**, wherein the cache is a multiple set-associative cache, wherein the cache comprises a plurality of sets or ways, and wherein the locking cache comprises one or more sets or ways of the cache, but not all of the sets or ways.

12. The computer system of claim **11**, further comprising a replacement management table, wherein the entries in the replacement management table indicate which set or sets in a cache are available for a transaction.

13. A computer program product comprising a computer readable storage medium having a computer readable program, wherein the computer readable program, when executed on a computing device, causes the computing device to:

issue a store instruction from a processor within the computing device, wherein the computing device comprises a cache that is partitioned into a locking cache and a non-locking cache, and wherein the locking cache is configured so that data written to the locking cache will not be overwritten until the data is loaded for use;

responsive to the data being written to the locking cache, issue a first signal to an I/O subsystem indicating that the data in the locking cache is ready for use by the I/O subsystem; and

receive a second signal indicating that the data in the locking cache can be overwritten responsive to the data being loaded for use by the I/O subsystem.

14. The computer program product of claim **13**, wherein the computer readable program, when executed on the computing device, further causes the computing device to:

issue a load request without intent to modify for the data.

15. The computer program product of claim **13**, wherein the computer readable program, when executed on the computing device, further causes the computing device to:

mark the data in the locking cache as "valid and modified".

16. The computer program product of claim **13**, wherein the locking cache is partitioned into a first partition and a second partition, and wherein the computer readable program, when executed on the computing device, further causes the computing device to:

write data to the second partition after the completion of writing data to the first partition.

17. The computer program product of claim **13**, wherein the computing device comprises one or more address range registers that store a specified address range to access the locking cache such that the locking cache appears as additional system memory.

18. The computer program product of claim **17**, wherein a cache controller is configured to determine whether the store instruction is within the specified address range for the locking cache.

* * * * *